МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

дисциплина «Базы данных»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4

«Разработка пользовательских функций, хранимых процедур, триггеров, курсоров, инструкций управления транзакциями»

Вариант №5

Выполнил: студент гр. ИТИ-22

Карпенко Д.Е.

Принял: преподаватель-стажёр

Карась О.В.

Гомель 2023

**Цель работы:** изучение возможностей основных программных модулей (функций, хранимых процедур) СУБД *MS SQL Server* и управления процессами изменения данных.

**Задание:**

**Разработка пользовательских функций**

В ранее разработанной БД *MS* *SQL Server* "Платежи студентов" необходимо создать пользовательские функции – по одной функции каждого из типов:

3.1.1 скалярные функции;

3.1.2 табличные функции *Inline*;

3.1.3 табличные функции *Multistatement*.

*3.1* Создать пользовательскую скалярную функцию, которая будет возвращать:

количество платежей для конкретного города (параметр – код города).

*3.2* Создать пользовательскую табличную функцию *Inline*, которая будет возвращать следующую информацию:

телефоны студентов, производивших платежи в заданный период времени.

*3.3* Создать пользовательскую табличную функцию *Multistatement*, которая будет выполнять следующую обработку данных:

вывести сведения о студентах (ФИО, факультет, дата рождения), достигших 18 лет в текущем семестре, перенести в таблицу с заданным именем.

**Разработка хранимых процедур**

В ранее разработанной БД *MS* *SQL Server* "Платежи студентов" создать хранимые процедуры.

(***Все процедуры должны содержать входные и выходные параметры,***

***значения параметров по умолчанию***)

которая выводит сведения о студентах заданного факультета (входной параметр), у которых сумма оплат ниже заданной (входной параметр).

**Разработка триггеров**

В ранее разработанной БД *MS SQL Server* "Платежи студентов" создать триггеры.

3.2.1 *DML*- триггеры типа *AFTER*

3.2.2 *DML*- триггеры типа *INSTEAD OF*

3.3.3 *DDL*-триггеры.

**Вариант задания на создание *DML*-триггеров типа *AFTER***

Отправка клиенту сообщения при попытке вставить данные в таблицу Студенты с некорректным годом поступления.

**Вариант задания на создание *DML*-триггеров типа *INSTEAD OF***

Отменить удаление строки из таблицы Студенты, если имеются связанные строки в таблице Платежи.

**Вариант задания на создание *DDL*-триггеров**

Соответствующего сообщения.

Создать *DDL* триггер, запрещающий изменять представления в БД с выводом

**3.4. Разработка курсоров**

В ранее разработанной БД *MS SQL Server* "Платежи студентов" создать курсоры:

*3.4.1* динамический;

*3.4.2* для выборки данных;

*3.4.3* для модификации данных.

Самостоятельно определить таблицы, над данными которых будут производиться манипуляции, назначение и выполняемые операции создаваемых курсоров; протестировать их работу.

**Ход работы**

На рисунке 1 представлен скалярная *SQL* функция.

*DECLARE* – Создание переменных;

*CREATE FUNCTION* – Создание функции;

*BEGIN…END* – Тело функции.

*RETURN* – Возвращающее значение

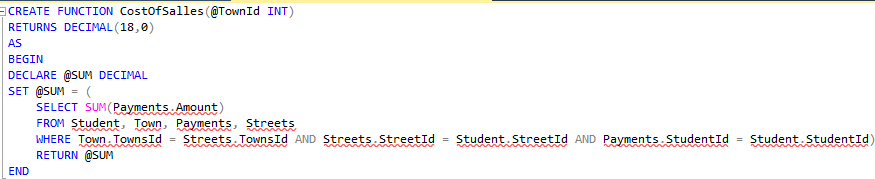


Рисунок 1 – Скалярная функция

На рисунке 2 представлен *SQL* запрос на вызов скалярной функции.

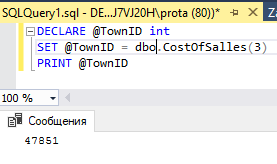


Рисунок 2 – Вызов скалярной функции

На рисунке 3 представлен табличная SQL функция *Inline*.

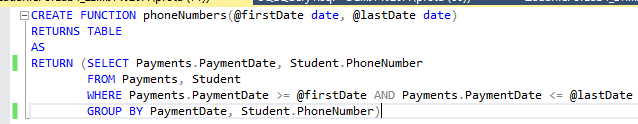


Рисунок 3 – Табличная функция *Inline*

На рисунке 4 представлен вызов табличной функции *Inline*.

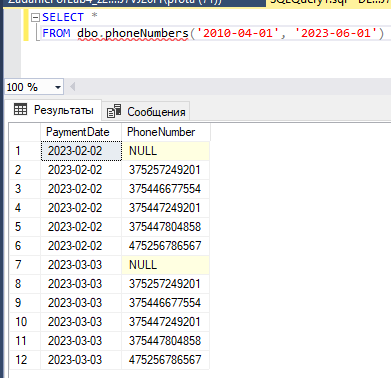
****

Рисунок 4 – Вызов табличной функции *Inline*

На рисунке 5 представлена табличная функция*Multistatement*.

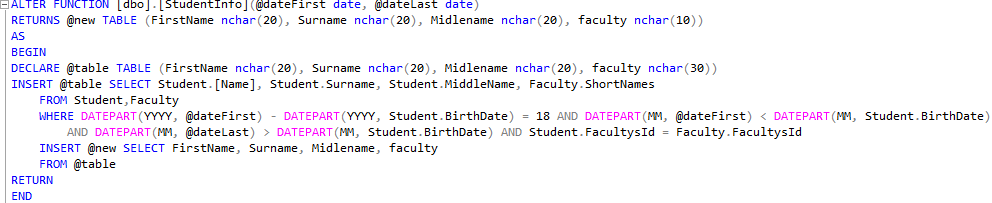
****

Рисунок 5 **-** Табличная функции *Multistatement*

На рисунке 6 представлен фрагмент с данными таблицы и вызов функции для проверки результатов.

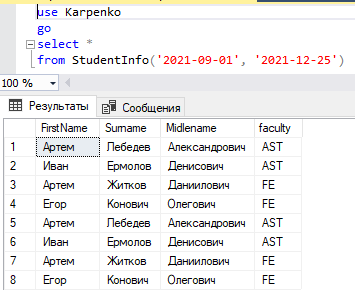
****

Рисунок 6 – Вызов табличной функции *Multistatement*

На рисуноке 7 представлена хранимая процедура с параметрами по умолчанию и входными параметрами @*old* и @*new*.

*CREATE PROCEDURE* – Создание хранимой процедуры.

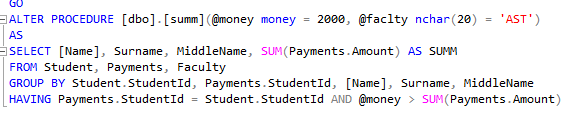


Рисунок 7 – Хранимая процедура

На рисунке 8 представлена функция вызова хранимой процедуры.

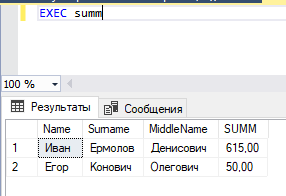
****

Рисунок 8 – Вызов хранимой процедуры

На рисунке 9 представлен *DML*-триггер тип *AFTER*.

*CREATE TRIGGER* – Создание триггера.

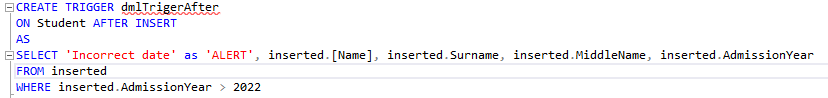
****

Рисунок 9 – *DML*-триггер

На рисунке 10 представлена работа *DML*-триггера типа *AFTER*.

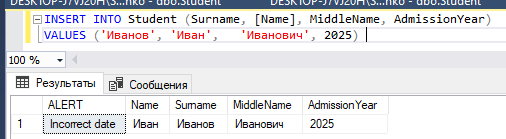
****

Рисунок 10 – Проверка *DML*-триггера типа *AFTER*

На рисунке 11 представлен *DML*-триггер типа *INSTEAD OF*.

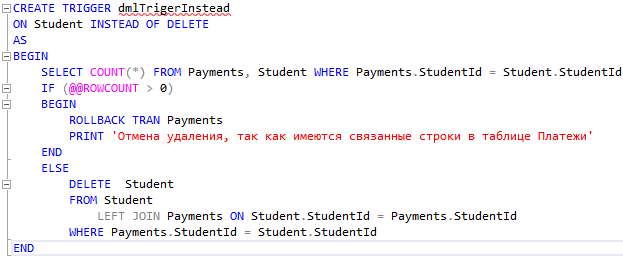
****

Рисунок 11 – *DML*-триггер типа *INSTEAD OF*

На рисунке 12 представлена работа *DML*-триггера типа *INSTEAD OF* c генерацией ошибки при попытке изменения таблицы.

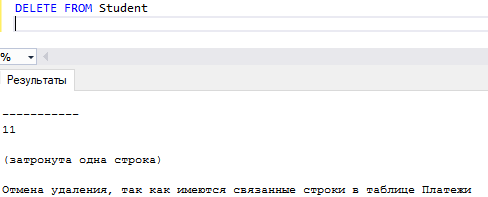
****

Рисунок 12 – Работа *DML*-триггера типа *INSTEAD OF*

На рисунке 13 представлен DDL-триггер на блокирования создания новых таблиц в базы данных.

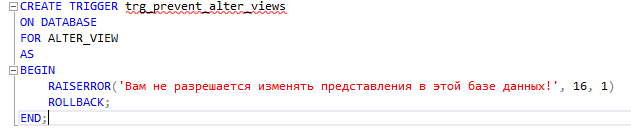


Рисунок 13 – *DDL*-триггер

На рисунке 14 представлена проверка *DDL*-триггера

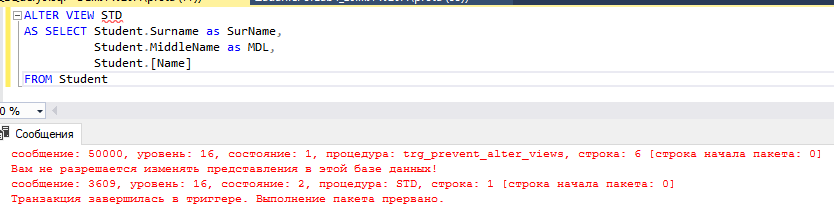


Рисунок 14 – Проверка *DDL*-триггера

На рисунке 15 представлен динамический курсор для выборки данных.

*OPEN* – Открытие курсора;

*FETH NEXT FROM* – Передвижение по курсору;

@@*FETH\_STATUS* – Функция возвращения статуса последнего курсора;

*CLOSE* – Зкарытие курсора;

*DEALLOCALE* – Удаление курсора.

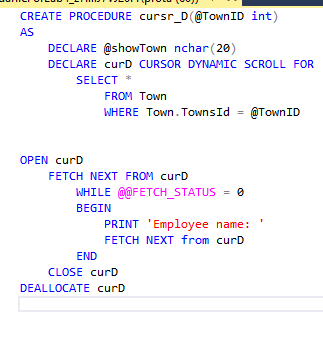
****

Рисунок 15 – Динамический курсор с выборкой данных

На рисунке 16 представлено выполнение динамического курсора с выборкой данных.

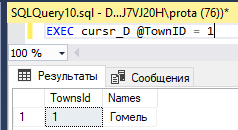
****

Рисунок 16 – Выполнение динамический курсор с выборкой данных

На рисунке 17 представлен курсор с модификацией данных.

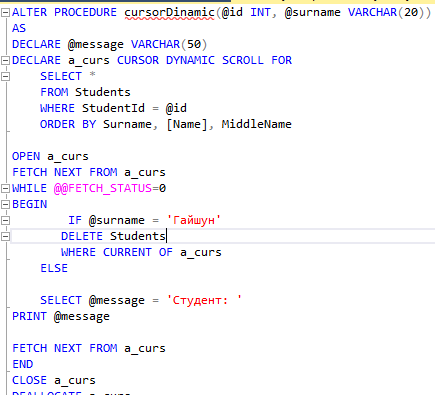
****

Рисунок 17 – Курсор с модификацией данных

На рисунке 18 представлен пример выполнения курсора с модификацией данных.

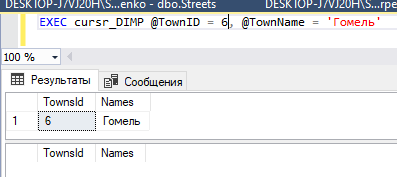
****

Рисунок 18 – Пример курсора с модификацией данных

**Вывод**: в результате выполнения лабораторной работы были получены знания в создании и реализации основных программных модулей (функций, триггеров, хранимых процедур, курсоров).

**ПРИЛОЖЕНИЕ А**

**Текст скриптов**

**Файл *Скалярная.sql*:**

CREATE FUNCTION CostOfSalles(@TownId INT)

RETURNS DECIMAL(18,0)

AS

BEGIN

DECLARE @SUM DECIMAL

SET @SUM = (

SELECT SUM(Payments.Amount)

FROM Student, Town, Payments, Streets

WHERE Town.TownsId = Streets.TownsId AND Streets.StreetId = Student.StreetId AND Payments.StudentId = Student.StudentId)

RETURN @SUM

END

**Файл *Табличная.sql*:**

CREATE FUNCTION phoneNumbers(@firstDate date, @lastDate date)

RETURNS TABLE

AS

RETURN (SELECT Payments.PaymentDate, Student.PhoneNumber

FROM Payments, Student

WHERE Payments.PaymentDate >= @firstDate AND Payments.PaymentDate <= @lastDate

GROUP BY PaymentDate, Student.PhoneNumber)

**Файл *MultiTable.sql*:**

USE [Karpenko]

GO

/\*\*\*\*\*\* Object: UserDefinedFunction [dbo].[StudentInfo] Script Date: 25.03.2023 13:16:42 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER FUNCTION [dbo].[StudentInfo](@dateFirst date, @dateLast date)

RETURNS @new TABLE (FirstName nchar(20), Surname nchar(20), Midlename nchar(20), faculty nchar(10))

AS

BEGIN

DECLARE @table TABLE (FirstName nchar(20), Surname nchar(20), Midlename nchar(20), faculty nchar(30))

INSERT @table SELECT Student.[Name], Student.Surname, Student.MiddleName, Faculty.ShortNames

FROM Student,Faculty

WHERE DATEPART(YYYY, @dateFirst) - DATEPART(YYYY, Student.BirthDate) = 18 AND DATEPART(MM, @dateFirst) < DATEPART(MM, Student.BirthDate)

AND DATEPART(MM, @dateLast) > DATEPART(MM, Student.BirthDate) AND Student.FacultysId = Faculty.FacultysId

INSERT @new SELECT FirstName, Surname, Midlename, faculty

FROM @table

RETURN

END

**Файл *Procedure.sql*:**

USE [Karpenko]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[summ] Script Date: 25.03.2023 16:01:59 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[summ](@money money = 2000, @faclty nchar(20) = 'AST')

AS

SELECT [Name], Surname, MiddleName, SUM(Payments.Amount) AS SUMM

FROM Student, Payments, Faculty

GROUP BY Student.StudentId, Payments.StudentId, [Name], Surname, MiddleName

HAVING Payments.StudentId = Student.StudentId AND @money > SUM(Payments.Amount)

**Файл *DMLTriggerAfter.sql*:**

CREATE TRIGGER dmlTrigerAfter

ON Student AFTER INSERT

AS

SELECT 'Incorrect date' as 'ALERT', inserted.[Name], inserted.Surname, inserted.MiddleName, inserted.AdmissionYear

FROM inserted

WHERE inserted.AdmissionYear > 2022

**Файл *DMLTriggerInstead.sql:***

CREATE TRIGGER dmlTrigerInstead

ON Student INSTEAD OF DELETE

AS

BEGIN

SELECT COUNT(\*) FROM Payments, Student WHERE Payments.StudentId = Student.StudentId

IF (@@ROWCOUNT > 0)

BEGIN

BEGIN TRANSACTION Payments

ROLLBACK TRANSACTION

PRINT 'Отмена удаления, так как имеются связанные строки в таблице Платежи'

END

ELSE

COMMIT TRANSACTION

DELETE Student

FROM Student

LEFT JOIN Payments ON Student.StudentId = Payments.StudentId

WHERE Payments.StudentId = Student.StudentId

END

**Файл *DDLTrigger.sql:***

CREATE TRIGGER trg\_prevent\_alter\_views

ON DATABASE

FOR ALTER\_VIEW

AS

BEGIN

RAISERROR('Вам не разрешается изменять представления в этой базе данных!', 16, 1)

ROLLBACK;

END;

**Файл** ***Cursor.sql:***

CREATE PROCEDURE cursr\_D(@TownID int)

AS

DECLARE @showTown nchar(20)

DECLARE curD CURSOR DYNAMIC SCROLL FOR

SELECT \*

FROM Town

WHERE Town.TownsId = @TownID

OPEN curD

FETCH NEXT FROM curD

WHILE @@FETCH\_STATUS = 0

BEGIN

PRINT 'Employee name: '

FETCH NEXT from curD

END

CLOSE curD

DEALLOCATE curD

**Файл ZadanieForLab4\_z8.sql**

CREATE PROCEDURE cursr\_DIMP(@TownID INT, @TownName VARCHAR(20))

AS

DECLARE @showTown nchar(20)

DECLARE curD CURSOR DYNAMIC SCROLL FOR

SELECT \*

FROM Town

WHERE Town.TownsId = @TownID

OPEN curD

FETCH NEXT FROM curD

WHILE @@FETCH\_STATUS=0

BEGIN

IF @TownName = 'Гомель'

DELETE Town

WHERE CURRENT OF curD

ELSE

SELECT @showTown = 'Город: '

PRINT @showTown

FETCH NEXT FROM curD

END

CLOSE curD

DEALLOCATE curD